UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERIA

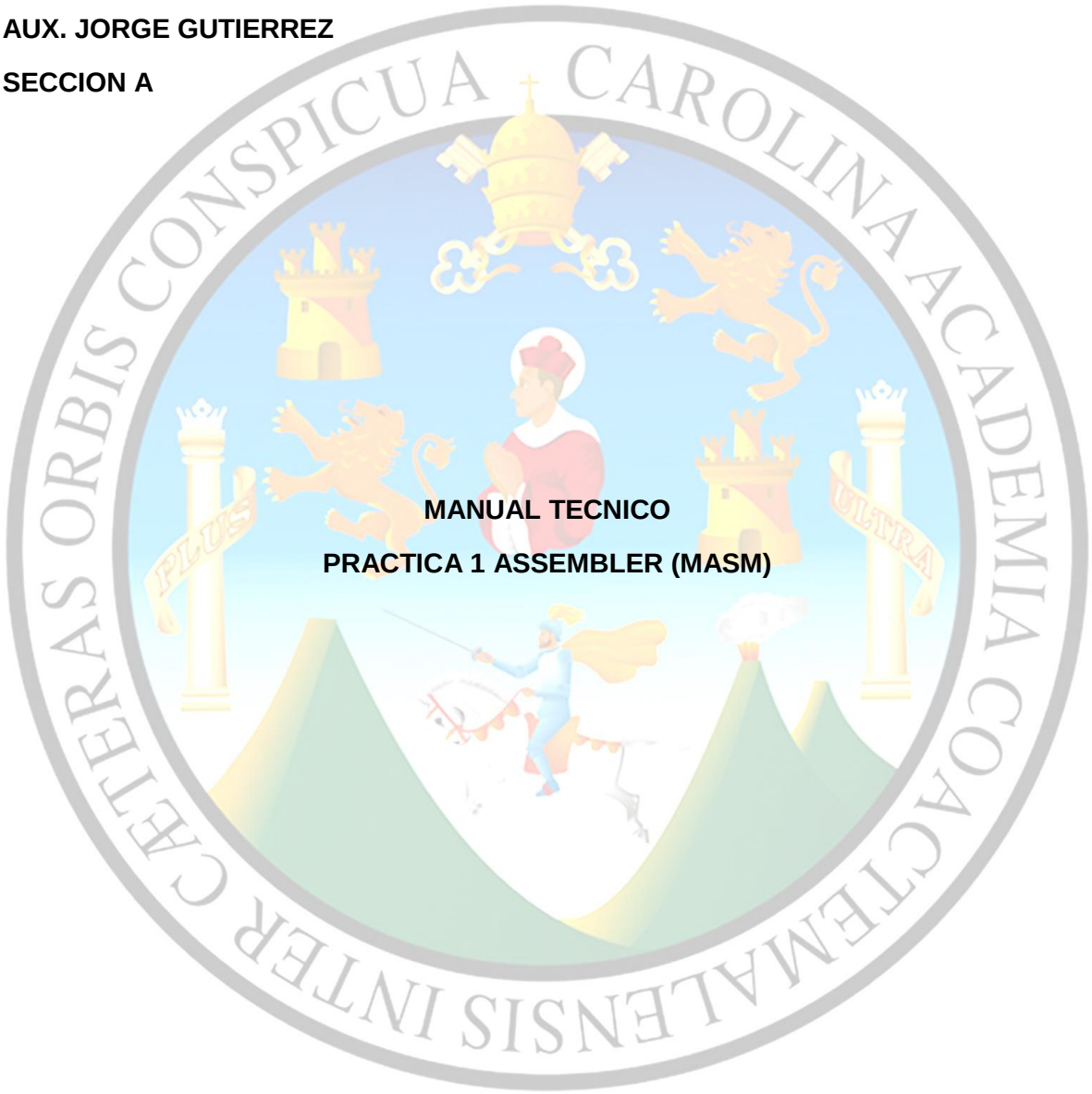
CIENCIAS Y SISTEMAS

ARQUITECTURA DE COMPUTADORES 1

ING. OTTO RENE LEIVA

AUX. JORGE GUTIERREZ

SECCION A



MANUAL TECNICO

PRACTICA 1 ASSEMBLER (MASM)

NOMBRE: OSCAR RENE CUELLAR MANCILLA

CARNET: 201503712

FECHA: 8 DE SEPTIEMBRE DE 2017

ENSAMBLADOR USADO EN LA PRACTICA: (MASM)

El Microsoft Macro Assembler (MASM) es un ensamblador para la familia x86 de microprocesadores. Fue producido originalmente por Microsoft para el trabajo de desarrollo en su sistema operativo MS-DOS, y fue durante cierto tiempo el ensamblador más popular disponible para ese sistema operativo. El MASM soportó una amplia variedad de facilidades para macros y programación estructurada, incluyendo construcciones de alto nivel para bucles, llamadas a procedimientos y alternación (por lo tanto, MASM es un ejemplo de un ensamblador de alto nivel). Versiones posteriores agregaron la capacidad de producir programas para los sistemas operativos Windows. MASM es una de las pocas herramientas de desarrollo de Microsoft para las cuales no había versiones separadas de 16 bits y 32 bits.

INTERRUPCIONES USADAS EN LA PRACTICA:

Interrupción al sistema 21H:

La mayoría de servicios ó funciones del sistema operativo MS-DOS se obtienen a través de la interrupción software 21H. Es por esto que se le denomina DOS-API: DOS-APPLICATION-PROGRAM-INTERFACE La INT 21H está compuesta por un grupo de funciones. Cuando se accede a la INT 21H, hay que indicar el número de función que queremos ejecutar. La llamada a la INT 21H se realizará como sigue:

- Introducimos en (AH) el número de función a la que deseamos acceder.

- En caso de que deseemos acceder a una sub-función dentro de una función, debemos indicarlo introduciendo en (AL) el número de esa sub-función.

- Llamar a la INT 21H.

CÓDIGO RELEVANTE DE LA PRACTICA:

Durante la realización de la práctica se utilizaron varios macros, el más utilizado sería el macro llamado “print” con el cual mando como parámetro una cadena a imprimir, se manda al registro AX el @data que representa que se va a escribir una cadena, con el mov ah,09 le indico a mi interrupción 21H que iniciare una impresión de cadena en pantalla y al registro DX le mando la dirección donde se almacena mi cadena a imprimir

```
print macro cadena
  mov ax,@data
  mov ds,ax
  mov ah,09
  mov dx,offset cadena
  int 21h
endm
```

También se crearon los macros para el manejo de los archivos, con nombres específicos que le dan un apego al lenguaje C.

```
fopen macro ruta,handle
print ruta
lea dx,ruta
mov ah,3dh
mov al, 00h
int 21h
mov handle,ax
jc NoSeEncontroRuta
endm

fwrite macro numbytes,databuffer,handle
mov ah,40h
mov bx,handle
mov cx,numbytes
lea dx,databuffer
int 21h
endm

fread macro numbytes,databuffer,handle
mov ah,3fh
mov bx,handle
mov cx,numbytes
lea dx,databuffer
int 21h
jc ErrorAlLeer
endm

fclose macro handle
mov ah,3eh
mov bx,handle
int 21h
jc ErrorAlCerrar
endm
```

En estos macros se maneja un handle, que es un número que representa de manera única a un archivo para poder ser manejado de una manera más accesible. Para entrar más a detalle acerca de los valores que reciben las interrupciones para poder realizar el manejo de archivos consultar la siguiente página donde se documenta que valores devuelve y pide cada función dada en el registro alto de AX.

Pagina:

http://ict.udlap.mx/people/oleg/docencia/ASSEMBLER/asm_interrup_21.html

Se declararon todos los mensajes que podían ser mostrados en consola con variables que los representaban de tipo byte:

```
mov1 db 0ah,0dh, '  haciendo division', '$'
mov2 db 0ah,0dh, '  comprobando split', '$'
mov3 db 0ah,0dh, '  no es split', '$'
err0 db 0ah,0dh, '  SEPARADOR INCORRECTO -> ', '$'
err1 db 0ah,0dh, '  CARACTER INVALIDO -> ', '$'
err2 db 0ah,0dh, '  FALTA SALTO DE LINEA (\n).............', '$'
msg1 db 0ah,0dh, '  REPORTE CREADO CON EXITO..............', '$'
msg2 db 0ah,0dh, '  NO SE CREO EL ARCHIVO REPORTE.........', '$'
msg3 db 0ah,0dh, '  NO SE ENCONTRO EL ARCHIVO.............', '$'
msg4 db 0ah,0dh, '  NO SE LEYO BIEN EL ARCHIVO............', '$'
msg5 db 0ah,0dh, '  NO SE CERRO BIEN EL ARCHIVO...........', '$'
msg6 db 0ah,0dh, '  ARCHIVO LEIDO Y SIN ERRORES...........', '$'
msg7 db 0ah,0dh, '  LISTA NUMEROS ESPEJO..................', '$'
msg8 db 0ah,0dh, '  LISTA NUMEROS PALINDROMOS.............', '$'
msg9 db 0ah,0dh, '  LISTA NUMEROS PARES...................', '$'
ms10 db 0ah,0dh, '  LISTA NUMEROS IMPARES.................', '$'
ms11 db 0ah,0dh, '  LISTA NUMEROS SPLIT...................', '$'
enc0 db 0ah,0dh, '  UNIVERSIDAD DE SAN CARLOS DE GUATEMALA', '$'
enc1 db 0ah,0dh, '  FACULTAD DE INGENIERIA', '$'
enc2 db 0ah,0dh, '  ESCUELA DE CIENCIAS Y SISTEMAS', '$'
enc3 db 0ah,0dh, '  ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1 A', '$'
enc4 db 0ah,0dh, '  SEGUNDO SEMESTRE 2017', '$'
enc5 db 0ah,0dh, '  OSCAR RENE CUELLAR MANCILLA', '$'
enc6 db 0ah,0dh, '  201503712', '$'
enc7 db 0ah,0dh, '  REPORTE PRACTICA NO. 1', '$'
enc8 db 0ah,0dh, '  FECHA 8 DE SEPTIEMBRE DE 2017', '$'
lin0 db 0ah,0dh, '  &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&', '$'
lin1 db 0ah,0dh, '  &&&&&&&&&& MENU PRINCIPAL &&&&&&&&&&', '$'
lin2 db 0ah,0dh, '  &&&&&&  1. CARGAR ARCHIVO   &&&&&&&&', '$'
lin3 db 0ah,0dh, '  &&&&&&  2. SALIR            &&&&&&&&', '$'
lin4 db 0ah,0dh, '  &&&&& INGRESE RUTA DEL ARCHIVO &&&&&', '$'
ope0 db 0ah,0dh, '  &&&&&  1. OBTENER PARES        &&&&&', '$'
ope1 db 0ah,0dh, '  &&&&&  2. OBTENER IMPARES      &&&&&', '$'
ope2 db 0ah,0dh, '  &&&&&  3. OBTENER ESPEJOS      &&&&&', '$'
ope3 db 0ah,0dh, '  &&&&&  4. OBTENER PALINDROMOS  &&&&&', '$'
ope4 db 0ah,0dh, '  &&&&&  5. OBTENER SPLIT        &&&&&', '$'
ope5 db 0ah,0dh, '  &&&&&  6. LISTA ASCENDENTE     &&&&&', '$'
ope6 db 0ah,0dh, '  &&&&&  7. LISTA DESCENDENTE    &&&&&', '$'
ope7 db 0ah,0dh, '  &&&&&  8. CREAR REPORTE        &&&&&', '$'
ope8 db 0ah,0dh, '  &&&&&  9. SALIR                &&&&&', '$'
```

Cada uno de los cuales es llamado en diferente ocasión y con diferente propósito.

Para obtener la ruta que ingresa el usuario desde la consola se utiliza la siguiente función:

```
Obtener_Ruta:
    mov ah,01h
    int 21h
    cmp al,26h ;ascii del &
    je Obtener_Ruta
    cmp al,0dh ;ascii del \n
    je FinRuta
    mov ArrayRuta[si],al
    inc si
    jmp Obtener_Ruta
```

En ella leo carácter por carácter que se va ingresando, primero comparo si el carácter es igual a (&), si es igual lo omito y sigo leyendo los demás caracteres, luego si el carácter es un salto de línea es porque el usuario oprimió enter para finalizar el ingreso de la ruta entonces llamo al método FinRuta para salir del ciclo, si aún no se ha presionado enter, entonces guardo el carácter en un arreglo que me representa a mis caracteres de la ruta y aumento el registro de desplazamiento (si) en uno.

Búsqueda de errores en el archivo: para ella realizo el siguiente algoritmo:

```
;--------------METODO PARA ANALIZAR EL CONTENIDO DEL ARCHIVO EN BUSCA DE ERRORES-----------
Analizar_Archivo:
    xor si,si;dejo en 0 el apuntador
    jmp PrimerSeparador
OmitirSaltos:
    inc si
    inc si
    ;impCad mov0
PrimerSeparador:
    cmp ArrayInfo[si],2Bh ;43 ascii del +
    jne FaltaSimbolo
    inc si
    ;impCad mov1
SigueNumero:
    cmp ArrayInfo[si],30h ;48 ascii del numero 0
    jb  CaracterInvalido ; si es menor al ascii del numero 0 es por que es un error de CaracterInvalido
    cmp ArrayInfo[si],39h ;57 ascii del numero 9
    ja CaracterInvalido ; si es mayor que el ascii del numero 9 tambien es un error de caracter
    inc si
    ;impCad mov2
    cmp ArrayInfo[si],2Bh ;43 ascii del +
    jne SigueNumero ;si no son iguales es por que aun no ha terminado el numero y se sigue leyendo
    inc si
    ;impCad mov3
    cmp ArrayInfo[si],3Bh ;59 ascii del ;
    je Menu_Operaciones1
    cmp ArrayInfo[si],0dh ;salto de linea
    je OmitirSaltos
    jmp FaltaSalto
;-------------------------------------METODO PARA BUSCAR LOS NUMEROS ESPEJO-----------------------------
```

El algoritmo anterior pretende leer carácter por carácter el arreglo donde tengo la información del archivo de entrada guardada y realiza un pequeño scanner en el cual espera primero un (+), seguido de los números que sean necesarios y que finalice nuevamente con un (+) si luego del (+) se lee un (\n) es porque sigue leyendo más números y si lo que encuentra es un (;) es porque la lectura llego a su fin y no hubo ningún error, si el análisis es pausado en algún momento se manda a llamar a las etiquetas que muestran el error y mostrando que fue lo que lo produjeron.

Se utilizo una plantilla que tiene como base el analizador para realizar todas las búsquedas que se solicitaban en la práctica como lo son: Números espejo, palíndromos, Split etc… Con ella se sabe en que momento del análisis se encuentra un número para poder realizar posteriores acciones con él.

```
----------PLANTILLA-----------------
    ImprimirSplit:
        jmp Esperar_Enter_Operaciones
    BuscarSplit:
        xor si,si;dejo en 0 el apuntador
        xor di,di;dejo en 0 el apuntador
        jmp VieneMas4
    Saltos4:
        inc si
        inc si
    VieneMas4:
        inc si
    VieneNumero4:
        ;--------BLOQUE DONDE VIENE NUMERO-----------
        ;-----------------------------------------
        inc si
        cmp ArrayInfo[si],2Bh ;ascii del +
        jne VieneNumero4 ;si no viene un mas es por que sigue viniendo un numero
    FinNumero4:
        inc si
        cmp ArrayInfo[si],3Bh ;59 ascii del ;
        je ImprimirSplit;imprimo los numeros en espejo
        cmp ArrayInfo[si],0dh ;salto de linea
        je Saltos4
```